AFRL-AFOSR-VA-TR-2015-0266

Discovering and Analyzing Network Function and Structure

Daniel Spielman
YALE UNIV NEW HAVEN CT

07/08/2015
Final Report

Air Force Research Laboratory
AF Office Of Scientific Research (AFOSR)/ RTA2
Arlington, Virginia 22203
Air Force Materiel Command

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
22-06-2015	Final	01-04-2012 to 31-03-2015

4. TITLE AND SUBTITLE
Discovering and Analyzing Network Function and Structure

5a. CONTRACT NUMBER
FA9550-12-1-0175

5b. GRANT NUMBER
FA9550-12-1-0175

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Spielman, Daniel A.

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Yale University
47 College Street, Suite 203
New Haven, CT 06510-3209

8. PERFORMING ORGANIZATION REPORT NUMBER
L00119/12-002832

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Kenechia Clarke-Day, Manager Financial Reporting and Analysis
Grant and Contract Financial Administration Yale University
47 College Street, Suite 216
New Haven, CT 06510-3209

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
L00119

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION A

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This award has supported the development of new approaches and algorithms for inference problems on networks. These have been validated by experiments demonstrating their utility in detecting spam web pages. It has also supported the development of faster algorithms for determining which edges are most critical to the structure of a network.

15. SUBJECT TERMS
Network Algorithms, Spam Detection, Learning on Networks, Critical Link Detection

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE	None		Spielman, Daniel A.
					19b. TELEPHONE NUMBER *(Include area code)* 203-436-1264

Reset

INSTRUCTIONS FOR COMPLETING SF 298

1. REPORT DATE. Full publication date, including day, month, if available. Must cite at least the year and be Year 2000 compliant, e.g. 30-06-1998; xx-06-1998; xx-xx-1998.

2. REPORT TYPE. State the type of report, such as final, technical, interim, memorandum, master's thesis, progress, quarterly, research, special, group study, etc.

3. DATES COVERED. Indicate the time during which the work was performed and the report was written, e.g., Jun 1997 - Jun 1998; 1-10 Jun 1996; May - Nov 1998; Nov 1998.

4. TITLE. Enter title and subtitle with volume number and part number, if applicable. On classified documents, enter the title classification in parentheses.

5a. CONTRACT NUMBER. Enter all contract numbers as they appear in the report, e.g. F33615-86-C-5169.

5b. GRANT NUMBER. Enter all grant numbers as they appear in the report, e.g. AFOSR-82-1234.

5c. PROGRAM ELEMENT NUMBER. Enter all program element numbers as they appear in the report, e.g. 61101A.

5d. PROJECT NUMBER. Enter all project numbers as they appear in the report, e.g. 1F665702D1257; ILIR.

5e. TASK NUMBER. Enter all task numbers as they appear in the report, e.g. 05; RF0330201; T4112.

5f. WORK UNIT NUMBER. Enter all work unit numbers as they appear in the report, e.g. 001; AFAPL30480105.

6. AUTHOR(S). Enter name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. The form of entry is the last name, first name, middle initial, and additional qualifiers separated by commas, e.g. Smith, Richard, J, Jr.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES). Self-explanatory.

8. PERFORMING ORGANIZATION REPORT NUMBER. Enter all unique alphanumeric report numbers assigned by the performing organization, e.g. BRL-1234; AFWL-TR-85-4017-Vol-21-PT-2.

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES). Enter the name and address of the organization(s) financially responsible for and monitoring the work.

10. SPONSOR/MONITOR'S ACRONYM(S). Enter, if available, e.g. BRL, ARDEC, NADC.

11. SPONSOR/MONITOR'S REPORT NUMBER(S). Enter report number as assigned by the sponsoring/ monitoring agency, if available, e.g. BRL-TR-829; -215.

12. DISTRIBUTION/AVAILABILITY STATEMENT. Use agency-mandated availability statements to indicate the public availability or distribution limitations of the report. If additional limitations/ restrictions or special markings are indicated, follow agency authorization procedures, e.g. RD/FRD, PROPIN, ITAR, etc. Include copyright information.

13. SUPPLEMENTARY NOTES. Enter information not included elsewhere such as: prepared in cooperation with; translation of; report supersedes; old edition number, etc.

14. ABSTRACT. A brief (approximately 200 words) factual summary of the most significant information.

15. SUBJECT TERMS. Key words or phrases identifying major concepts in the report.

16. SECURITY CLASSIFICATION. Enter security classification in accordance with security classification regulations, e.g. U, C, S, etc. If this form contains classified information, stamp classification level on the top and bottom of this page.

17. LIMITATION OF ABSTRACT. This block must be completed to assign a distribution limitation to the abstract. Enter UU (Unclassified Unlimited) or SAR (Same as Report). An entry in this block is necessary if the abstract is to be limited.

Final report for
"Discovering and Analyzing Network Function and Structure"
FA955012-10175
Daniel A. Spielman
June 22, 2015

The advances supported by this award may be roughly divided into three categories:

1. the development of faster algorithms for the standard formulation of the problem of interpolation on networks,

2. the development of a new approach to interpolation on networks, and

3. the development of faster algorithms for sparsifying and detecting critical edges in networks.

This report begins with a brief explanation of the problem of interpolation on networks, followed by an explanation of these advances.

1 Interpolation on Networks

In network interpolation problems, one is given a network along with information about some of the nodes in the network. Assuming that nodes that are connected by edges are similar, one is asked to guess the corresponding information for the remaining nodes. A benchmark example of such a problem is that of detecting spam webpages [CDB+06]. This problem arose in an effort at Microsoft to detect spam webpages that are created solely to increase the ranking of the pages they point to. The network in this problem has one vertex for every webpage, and edges representing the links between the webpages. Researchers at Microsoft investigated many webpages, and determined for each whether it was spam or legitimate. The interpolation problem is to use these determinations along with the link structure to estimate the likelihood that other webpages are spam. This is possible because legitimate webpages are unlikely to link to spam web pages, so a link to a spam web page is evidence of spam. Similarly, a link from a legitimate webpage is evidence of legitimacy.

Network interpolation problems arise in many other contexts in Machine Learning. One of the most famous examples comes from the work of Zhu, Ghahramani and Lafferty [ZGL+03] who showed how to convert many problems of classification and regression in Machine Learning into problems of interpolation on networks.

2 Faster Algorithms for the Standard Interpolation

The standard method of performing interpolation in networks, which we henceforth call l_2 minimization, comes from Zhu, Ghahramani and Lafferty [ZGL+03], and only applies to undirected networks. Formally, one is given a network with vertex set V and edge set E, along with a subset $S \subset V$ at which the values of the function f to be interpolated are known. The interpolation is performed by finding the function g that agrees with f on the vertices in S and minimizes the sum of the squares of the differences across edges:

$$\sum_{(u,v)\in E} w_{u,v} \left(g(u) - g(v)\right)^2, \tag{1}$$

where $w_{u,v}$ is the weight of edge (u, v).

The problem of finding this function g may be reduced to the problem of solving a system of linear equations in the Laplacian matrix of the network. In fact, this problem is mathematically identical to many problems that arise in Computational Science, including the computation of electrical flow and heat flow.

This award supported the development of three new algorithms for solving this problem. The first of them appears in the papers [CKP$^+$14, CKM$^+$14]. The randomized algorithm in this paper solves these problems to accuracy ϵ in networks with m edges in expected time $O(m\sqrt{\log m}\log 1/\epsilon)$. This is absurdly fast: for moderate ϵ it is less time than would be required to sort the weights of the edges in the network.

Given the speed of this algorithm, one may wonder why we would need another. The answer is that this algorithm is not well-suited to parallelization. This means that we do not know how to accelerate it substantially using multi-core processors, and that it is ill-suited for problems whose magnitude requires computation by clusters. For this reason we began the development of algorithms that have efficient parallel implementations.

Previously, all algorithms that solved Laplacian linear systems in nearly linear time employed two graph theoretic primitives: low stretch spanning trees and graph sparsifiers. While we had no idea how to compute low stretch spanning trees efficiently in parallel, we thought that it might be possible to compute graph sparsifiers this way. This motivated us to design an algorithm for solving Laplacian linear equations that only relies of graph sparsification. The resulting algorithm appeared in the paper [PS14].

This paper presented the first algorithm for solving Laplacian linear systems in polylogarithmic parallel time and nearly-linear work. That is, the algorithm requires little computation and is efficient in parallel. However, the algorithm in this paper is better viewed as a proof of concept than something that one should really implement: it requires time cubic in the logarithm of the condition number of the system to be solved. While this is asymptotically fast in theory, it is too slow for practical use. The advantage of the algorithm presented in this paper are that:

1. it introduced an entirely new approach to the fast solution of Laplacian linear systems,
2. this approach is very easy to understand,
3. it introduced the first efficiently parallelizable algorithms for graph sparsification, and
4. it inspired the development of even better parallel algorithms for graph sparsification [Kou14, CLM$^+$14].

The simplicity of the algorithm presented [PS14] has enabled us to improve it to obtain remarkably fast and efficient parallel algorithms in [LPS]. In this most recent work, we develop algorithms for solving Laplacian linear systems that run very quickly in parallel. They develop something that the numerical linear algebra community has been seeking for a long time: sparse approximate inverses. To explain these, I recall that the classical Gaussian Elimination algorithm for solving linear equations in a matrix A constructs triangular matrices L and U so that $LU = A$. One can then solve a system of linear equations in A by solving equations in L and U. This can be done quickly if L and U are sparse, as linear equations in triangular matrices can be solved with a number of computations proportional to their number of nonzero entries. We prove that every Laplacian matrix A has an approximate LU-factorization in which both L and U have $O(n)$

entries, where n is the dimension of A. This provides the first linear time sequential algorithm for approximately solving systems of equations in Laplacian matrices. Moreover, the matrices L and U we produce are very special: systems of equations in these matrices can be solved in parallel time $O(\log n \log\log n)$ and linear work. This is within an $O(\log\log n)$ factor of the best possible.

As is the case with Gaussian Elimination, it takes us longer to compute these matrices L and U than it does to use them to solve systems of linear equations. However, if we are willing to settle for slightly worse L and U, we can accelerate the computation. Our presently best algorithm computes the matrices L and U and applies them to solve a linear system in parallel time $O(\log^6 n)$. While this is still too slow to be practical, we are optimistic that it will eventually be possible to improve it to make it practical.

3 Sparsification and Significant Edge Detection

The fast parallel algorithms that we describe for solving systems of equations in Laplacian matrices requires fast parallel algorithms for network sparsification. Sparsification is the process of finding a sparse network that approximates a given network. We use the notion of spectral sparsification. Thus, the sparse networks we produce have approximately the same community structures as the original, and the solutions to interpolation problems in the sparse approximations are similar to those in the original (at least for the l_2 minimization described above).

Sparse approximations of networks are often diserable because they take less space to store. One may wonder why this is useful, as the networks we encounter are often sparse. The answer is that sparsification allows us to compactly store information about a network that can take a long time to compute. For example, we might want to keep track of all pairs of vertices that are within distance 2, 3, or even 4 of each other. In [PS14], we quickly compute sparsifiers that enable us to approximate all of these distance-k networks.

Our sparsification algorithms have two steps: we first assign a significance to every edge in a network, where we measure the significance of an edge by how useful it is to communication in a network. The most significant edges are those whose removal would disconnect the network. We then construct a sparse approximation of the network by randomly sampling the edges according to their significance. Thus, the first step of our algorithm is a fast parallel procedure for approximating the significance of every edge.

4 A Better Approach to Interpolation

The biggest advance supported by this award is the development of a new approach to performing interpolation in networks that we call "Lipschitz Learning" that overcomes three disadvantages of l_2 minimization:

1. l_2 minimization can only be applied in undirected networks,
2. empirically, l_2 minimization has been shown to have poor performance in large networks when the set of values at which the function is known is small, and
3. in l_2 minimization there is no easy way to compensate for errors in the edges of the network.

Lipschitz learning overcomes all three of these problems.

In our paper on Lipschitz learning [KRSS15a], we both define this new approach to interpolation on networks and develop reasonably fast algorithms. While these algorithms are not nearly as fast as those that we have developed for l_2 minimization, they are a good start: we can perform the interpolation on networks with millions of nodes in a few minutes. We have made an implementation of our algorithms available on GitHub [KRSS15b]. Our algorithms for dealing with errors in the input network and function values actually use the Laplacian linear system solvers.

4.1 Lipschitz Learning

There are three distinct ways of defining our approach to Lipschitz learning. The easiest way to think of it is that instead of minimizing (1), it begins by finding the function g that agrees with f on S that minimizes

$$\max_{(u,v)\in E} w_{(u,v)} \left| g(u) - g(v) \right|. \tag{2}$$

As this does not lead to a unique function g, we seek the function that minimizes the second-to-maximum of these quantities among those that minimize the maximum, and so on. The result is sometimes called the Absolutely Minimal Lipschitz Extension. We call it the *lex minimizer*.

Another way of defining it is to consider the problem of minimizing Laplacian p-norms introduced in [BZ13]:

$$\sum_{(u,v)\in E} \left(w_{u,v} \left| g(u) - g(v) \right| \right)^p, \tag{3}$$

over all functions that agree with f on g. The lex minimizer is the limit as p grows large of the function g that agrees with f on S that minimizes (3). However, we can compute the lex minimizer much faster than one can solve (3).

The third way of defining it is by analogy to one of the characterizations of the solution to (1). The minimizer of (1) for an unweighted network is the function g that agrees with f on S such that for every vertex not in S, the value of g at that vertex is the average of the value of g at its neighbors. In unweighted networks, the lex minimizer is the function g such that at every vertex not in S, the value of g is the average of the minimum and maximum values at its neighbors.

All these definitions can be naturally extended to directed networks.

4.2 Algorithms and Results

We observe experimentally that lex minimizers give much better predictions than l_2 minimization when the label set S is small. The most interesting example is the webspam data set, for which we obtain much better results than the previous algorithms [ZBT07].

One of the big advantages of lex minimizers over the standard approach of minimizing (1) is that they allow us to easily compensate for noise both in the values of f on S and in the actual edges of the network. We develop a fast algorithm to minimize (2) subject to a budget of changes to edge weights and values of f on S. We do this by formulating the resulting problem as a linear program, and by then designing a custom interior point method for solving the linear program. We prove that the interior point method requires few iterations, and that the dominant cost of each iteration is the solution of a system of linear equations in a Laplacian matrix.

We also show how to perform a rather surprising outlier removal in polynomial time: we can minimize (2) subject to the removal of a given number of vertices from S. That is, we can compensate for the possibility that some of the values are extremely wrong. While we do not yet know

a fast polynomial time algorithm for this task, we are optimistic that we may one day find one. It is particularly surprising that such an algorithm exists, as we show that the analogous problem for (1) is NP-complete.

5 Isotonic Regression

We have built on the techniques we developed in our work on Lipschitz learning to design the fastest algorithms for isotonic regression [KRS], a problem that has been studied since the 1950's. Isotonic regression problems are another type of inference problem on networks. They are specified by a directed acyclic network in which every node is associated with a real variable. The directed edges specify inequalities which it is known the nodes must satisfy: an edge from node u to node v indicates that the variable at node v must exceed the variable at node u. In addition, an estimate of every variable is provided. The isotonic regression problem is to compute values of the variables that come as close as possible to the given estimates subject to the inequalities dictated by the network.

Different measures of "close" provide very different computational problems. For measures of close in infinity norm or lexicographic infinity norm, we reduce the problem to that of computing lex minimizers. For l_p norms, we solve the problem by specially designed interior point methods. In fact, these are the first fast algorithms for the problem for p other than 1 and 2.

References

[BZ13] Nick Bridle and Xiaojin Zhu. p-voltages: Laplacian regularization for semi-supervised learning on high-dimensional data. In *Eleventh Workshop on Mining and Learning with Graphs (MLG2013)*, 2013.

[CDB+06] Carlos Castillo, Debora Donato, Luca Becchetti, Paolo Boldi, Stefano Leonardi, Massimo Santini, and Sebastiano Vigna. A reference collection for web spam. *SIGIR Forum*, 40(2):11–24, December 2006.

[CKM+14] Michael B. Cohen, Rasmus Kyng, Gary L. Miller, Jakub W. Pachocki, Richard Peng, Anup B. Rao, and Shen Chen Xu. Solving SDD linear systems in nearly $mlog^{1/2}n$ time. In *46th Annual ACM Symposium on Theory of Computing*, STOC '14, pages 343–352, 2014.

[CKP+14] Michael B. Cohen, Rasmus Kyng, Jakub W. Pachocki, Richard Peng, and Anup Rao. Preconditioning in expectation. *CoRR*, abs/1401.6236, 2014.

[CLM+14] Michael B Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. *arXiv preprint arXiv:1408.5099*, 2014.

[Kou14] Ioannis Koutis. Simple parallel and distributed algorithms for spectral graph sparsification. In *26th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '14, pages 61–66, New York, NY, USA, 2014. ACM.

[KRS] Rasmus Kyng, Anup Rao, and Sushant Sachdeva. Provable fast algorithms for isotonic regression in all norms. submitted to NIPS.

[KRSS15a] Rasmus Kyng, Anup Rao, Sushant Sachdeva, and Daniel A. Spielman. Algorithms for Lipschitz learning on graphs. In *Journal of Machine Learning Research*, 2015. to appear. Available at `http://arxiv.org/abs/1505.00290`.

[KRSS15b] Rasmus Kyng, Anup Rao, Sushant Sachdeva, and Daniel A. Spielman. YINSlex. `https://github.com/danspielman/YINSlex`, 2015.

[LPS] Yin-Tat Lee, Richard Peng, and Daniel A. Spielman. Sparsified cholesky solvers for sdd linear systems. in preparation.

[PS14] Richard Peng and Daniel A. Spielman. An efficient parallel solver for SDD linear systems. In *ACM Symposium on Theory of Computing*, pages 333–342, 2014.

[ZBT07] Dengyong Zhou, Christopher J. C. Burges, and Tao Tao. Transductive link spam detection. In *3rd International Workshop on Adversarial Information Retrieval on the Web*, AIRWeb '07, pages 21–28, New York, NY, USA, 2007. ACM.

[ZGL+03] Xiaojin Zhu, Zoubin Ghahramani, John Lafferty, et al. Semi-supervised learning using Gaussian fields and harmonic functions. In *Twentieth International Conference on Machine Learning*, pages 912–919, 2003.

1.

1. Report Type

Final Report

Primary Contact E-mail

Contact email if there is a problem with the report.

spielman@cs.yale.edu

Primary Contact Phone Number

Contact phone number if there is a problem with the report

203-436-1264

Organization / Institution name

Yale University

Grant/Contract Title

The full title of the funded effort.

Discovering and Analyzing Network Function and Structure

Grant/Contract Number

AFOSR assigned control number. It must begin with "FA9550" or "F49620" or "FA2386".

FA9550-12-1-0175

Principal Investigator Name

The full name of the principal investigator on the grant or contract.

Daniel A. Spielman

Program Manager

The AFOSR Program Manager currently assigned to the award

James Lawton

Reporting Period Start Date

04/01/2012

Reporting Period End Date

03/31/2015

Abstract

This award has supported the development of new approaches and algorithms for inference problems on networks. These have been validated by experiments demonstrating their utility in detecting spam web pages. It has also supported the development of faster algorithms for determining which edges are most critical to the structure of a network.

Distribution Statement

This is block 12 on the SF298 form.

Distribution A - Approved for Public Release

Explanation for Distribution Statement

If this is not approved for public release, please provide a short explanation. E.g., contains proprietary information.

SF298 Form

Please attach your SF298 form. A blank SF298 can be found here. Please do not password protect or secure the PDF

The maximum file size for an SF298 is 50MB.

Spielman_form298.pdf

Upload the Report Document. File must be a PDF. Please do not password protect or secure the PDF . The

maximum file size for the Report Document is 50MB.

finalReport.pdf

Upload a Report Document, if any. The maximum file size for the Report Document is 50MB.

Archival Publications (published) during reporting period:

Michael B. Cohen, Rasmus Kyng, Gary L. Miller, Jakub W. Pachocki, Richard Peng, Anup B. Rao, and Shen Chen Xu. Solving SDD linear systems in nearly mlog1/2n time. In 46th Annual ACM Symposium on Theory of Computing, STOC '14, pages 343–352, 2014.

Michael B. Cohen, Rasmus Kyng, Jakub W. Pachocki, Richard Peng, and Anup Rao. Preconditioning in expectation. CoRR, abs/1401.6236, 2014.

Rasmus Kyng, Anup Rao, Sushant Sachdeva, and Daniel A. Spielman. Algorithms for Lipschitz learning on graphs. In Journal of Machine Learning Research, 2015. to appear. Available at http://arxiv.org/abs/1505.00290.

Richard Peng and Daniel A. Spielman. An efficient parallel solver for SDD linear systems. In ACM Symposium on Theory of Computing, pages 333–342, 2014.

Changes in research objectives (if any):

Change in AFOSR Program Manager, if any:

The program manager is now James Lawton. When this award began, it was Robert Bonneau.

Extensions granted or milestones slipped, if any:

AFOSR LRIR Number

LRIR Title

Reporting Period

Laboratory Task Manager

Program Officer

Research Objectives

Technical Summary

Funding Summary by Cost Category (by FY, $K)

	Starting FY	FY+1	FY+2
Salary			
Equipment/Facilities			
Supplies			
Total			

Report Document

Report Document - Text Analysis

Report Document - Text Analysis

Appendix Documents

2. Thank You

E-mail user

Jun 29, 2015 11:16:04 Success: Email Sent to: spielman@cs.yale.edu